面向单指令集异构多核处理器的负载均衡策略

徐远超[1,2,3]，张志敏[1]，孙卫真[3]

(1. 中国科学院 计算技术研究所 计算机体系结构国家重点实验室（筹），北京 100190；
2. 中国科学院 研究生院，北京 100039；3. 首都师范大学 信息工程学院，北京 100048)

摘 要：不同的程序之间以及同一个大规模程序运行时的不同阶段常常表现出不同的行为，如果按需分配资源，异构多核比同构多核能表现出更好的性能功耗比。对同构和异构多核环境下的调度机制进行了比较，通过在内核中使用硬件性能计数器动态监测程序的行为和修改核的负载计算方法，实现了程序行为感知和异构感知的任务迁移规则。通过在DVFS技术构建的真实异构多核环境下测试，结果显示，不同属性的程序基本按照预定的理想调度方案进行任务分配和负载均衡。

关键词：调度；异构多核处理器；负载均衡；程序行为分析；硬件性能计数器

中图分类号：TP316 **文献标识码**：A **文章编号**：1000-436X(2011)9A-0204-07

Load balancing policy for single-ISA heterogeneous multi-core systems

XU Yuan-chao[1,2,3], ZHANG Zhi-min[1], SUN Wei-zhen[3]

(1. State Key Laboratory of Computer Architecture (Preparatory), Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China; 2. Graduate University of the Chinese Academy of Sciences, Beijing 100039, China; 3. College of Information Engineering, Capital Normal University, Beijing 100048, China)

Abstract: Different program or phase had different runtime behavior and resource requirement. If resource assignment on-demand was available on operating system scheduler, single-ISA asymmetric multi-core system can gain better performance per watt over symmetric multi-core system. Two kinds of scheduler between asymmetric and symmetric were compared. Through monitoring program phase behavior by using hardware performance counter and modifying the computation method of the load of each core, behavior-aware and heterogeneity-aware task moving rule was designed. Heterogeneous Multi-core system was constructed through dynamic voltage frequency scaling (DVFS). The experimental result shows that different tasks are moved to proper cores according to their behavior features when load is imbalance.

Key words: scheduling; heterogeneous multi-core; load balance; program behavior analysis; hardware performance counter

1 引言

由于时钟频率接近物理极限以及随之产生的温度过高等问题，现代微处理器设计不得不转变设计理念，从追求指令级并行的单核转向追求线程级并行的多核，从单一追求性能转向性能功耗比。

收稿日期：2011-07-05

基金项目：国家重点基础研究发展计划（“973”计划）基金资助项目（2011CB302501）；“核高基”国家科技重大专项基金资助项目（2011ZX01028-001-002）

Foundation Items: The National Basic Research Program of China (973 Program) (2011CB302501); The National Core-High Tech-Basic Program (2011ZX01028-001-002)

2003 年，美国加州大学圣迭戈分校的 R．Kumar 博士指出异构多核处理器是提高性能功耗比的有效方法[1]，仿真实验表明，同一个并行程序，在功耗与面积相同的前提下，在数量多的简单核上运行要比在数量少的复杂核上运行花费的时间短[2]，说明并行程序无法充分利用长流水线高指令级并行的复杂核上的资源，而更容易从高线程级并行度中受益。对于不同的程序，行为也不一样[3]，具体表现在微体系结构特征的多样性，即程序对资源的需求存在差异，有的以计算为主（cpu-intensive），有的以访存为主（mem-intensive），有的以 I/O 为主（I/O-intensive）。cpu-intensive 指程序具有很高指令级并行度和数据局部性，这类程序 CPU 利用率很高，适合在频率较高的核上运行；而 mem-intensive 和 I/O-intensive 由于访存和 I/O 延迟，使得 CPU 利用率不高，如果这类程序放在性能高的核上运行，绝对性能提高的同时功耗也在增加，如果放在性能稍低的核上运行，绝对的性能会下降，但功耗也在减少，如果用性能功耗比来衡量，后者要优于前者。在多核处理器中引入不对称性，正是迎合不同属性的应用，实现资源的按需分配。异构多核处理器有多种，本文特指单指令集异构多核处理器，因为其他异构多核处理器很难用操作系统来进行任务调度，并假定只有 2 种类型的核，简称为快核（fastcore）和慢核（slow core），这种假设并不影响研究结果的正确性，R．Kumar 通过实验得出[1]，核的种类 2 种最为适宜，否则，会使调度算法过于复杂、开销过大。

现有的操作系统在设计时都没有考虑异构，如 Linux，在启动时查询 0 号核（BP）的特征，然后直接将其应用于其他核（AP），只定义了一个变量 cpu_khz 记录所有核的频率。异构感知是有效调度的基础，由于核间性能的差异，核的负载需要定义新的计算公式，当负载均衡时，还要考虑目标核的属性与待迁移线程是否符合前面叙述的规则。

本文设计并实现了一个程序行为感知的负载均衡调度策略，通过硬件性能计数器获取程序行为特征，并修改了 CPU 负载的计算办法及任务迁移规则。本文的创新之处在于：基于真实的软硬件环境；在 Linux 内核中改写 Perfctr 驱动实现了程序行为的自动监测；对现有调度策略改动较小，较好地保持了原有调度策略的一些特征；无需定义阈值来区分程序类别，灵活性好。

2　研究现状

异构多核处理器应用的前提是程序行为感知的任务调度，问题的核心在于如何将程序行为分析和调度有机结合起来。程序行为有离线分析和在线分析 2 种。

R.Kumar 博士提出把程序在每个核上试探性运行一小段时间，根据试探运行期间的采样数据找出最优的核，从而把程序调度上去。显然，每一个程序都试探运行，开销太大[4]。D.Shelepov 提出一种基于体系结构签名（architectural signatures）的离线分析方法[5]，即在程序运行之前获取程序的属性，如是否访存密集、指令级并行度，然后注入到应用程序的二进制代码中，通过离线时收集的每个线程的体系结构特征，来指导调度[6]，该方法不需要试探运行，没有运行时在线分析的开销，但作者采用的是硬编码的方法，没有解决程序行为属性的二进制注入问题，使用的是用户级调度模拟器。

T.Sondag[7] 进一步提出程序段指导(phase-guided)的调度方法，可以针对不同的程序段行为进行更精确有效的调度，但作者提出的使用内嵌 PAPI 代码的方法并不简洁高效，PAPI 是方便程序员访问硬件性能计数器而提供的编程接口，开销较大，不适合在内核中使用，此外，文中提到的二进制注入只对单线程有效。

Tong LI 是较早在真实软硬件环境下开展异构多核调度的 Intel 研究人员[8]，通过修改 Linux 操作系统中与负载均衡相关的函数实现了异构感知，任务迁移时遵循“快核优先”的原则，在保证负载均衡的前提下，让快核充分利用。该方法没有考虑程序行为的差异。Koufaty 等人[9]改进了该算法，增加了程序行为分析，使用各种延迟来估算偏好值（bias），作为调度依据。但该算法需要使用 4 个硬件性能计数器，而一些处理器提供的硬件性能计数器少于 4 个。

A．Fedorova 研究小组给出了一个综合的调度算法，将效率和线程级同时加以考虑[10]。文中使用 Utility 因子和 Speedup 因子来指导调度，算法复杂，且只适合于 OpenMP 编程模型编写的并行程序，不适合使用 Pthread 库函数编写的多线程程序。

从文献资料来看，国内学者开展此方向研究的还不多见，仍然是沿用多处理器系统环境下以仿真为主的研究方法，由于多核环境下对开销十分敏

感，很多算法即使仿真时效果不错，但实现时却困难重重。

3 异构多核负载均衡策略的挑战

要实现异构多核环境下程序行为感知的负载均衡调度，要解决3个问题，第一问题是定义新的负载计算公式。由于核的频率、cache 大小以及其他体系结构特征不再完全相同，不能使用相同的标准计算负载，否则，无法显示出计算能力强弱的差异，从直觉上看，快核计算能力强大，理应承担更多的任务，负载的计算是负载均衡调度的基础。第二问题是硬件性能计数器引入的开销评估以及如何嵌入到内核中实现程序行为属性的自动监测。第三问题是在任务迁移时要考虑到任务属性的影响。

由于目前还没有商用的异构多核处理器，只能借助其他方法来构建异构多核环境。除此之外，要解决的关键技术在于程序行为分析技术如何与调度结合起来。常用的程序行为分析工具有 PIN、PAPI、Dyninst 等，但这些工具获取到的数据是面向用户层面的，如何记录这些数据以及如何将其传递给操作系统并与调度器无缝对接是最重要的挑战。程序行为离线分析能做到与微体系结构无关，具有更好的平台无关性，但很多属性只有在程序动态运行时才能监测到，如程序执行的时间、IPC、cache miss rate 等，更重要的是，离线分析还存在获取的数据如何记录以及如何传递给操作系统的问题；程序行为动态分析所获取的数据与体系结构相关，信息准确实时，但动态分析在程序执行时进行，是否会引入不可接受的开销，此外，动态分析也依赖于分析的工具，如硬件性能计数器，并非所有的属性都能动态监测到。

4 异构多核负载均衡策略的实现

4.1 程序行为动态分析

任务特性指该任务倾向于 cpu-intensive 或 mem-intensive。界定一个任务特性的标准有很多，如 cache miss rate、IPC、相对加速比等。本文使用 IPC 作为界定标准，原因在于 IPC 可以通过硬件性能计数器在任务运行过程中轻松获取，同时，对于仅频率不对称的异构多核处理器而言，从 SPEC CPU 2006 中随机选取了11个基准测试程序，发现 IPC 同 cache miss rate 以及加速比在走势上相似(如图1所示)，并且大部分程序的 IPC 基本不受 CPU 频率的影响（如图2所示），由于最优调度是 NP 难问题，因此 IPC 可以近似地作为程序类别的度量标准。当然，对于有更多方面不对称的复杂异构多核处理器而言，不能再使用单一的 IPC 作为度量标准，需要多种因素综合考虑，尚没有一致意见。

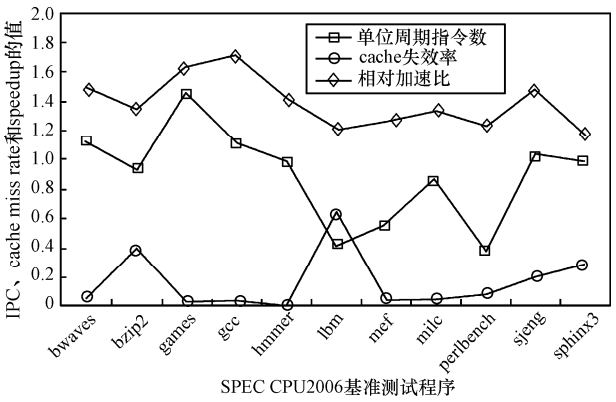

图1 IPC、cache miss rate 和 speedup 比较

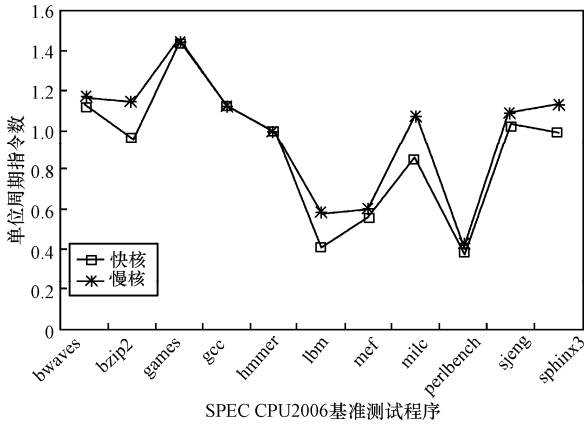

图2 benchmark 程序在快核与慢核的 IPC 比较

IPC 越大，一般情况下需要更多的计算资源，因此应该将这类任务分配给频率较高的核，反之，分配给频率较低的核，以避免功耗的浪费。

离线状态下任务的 IPC 可以借助 PAPI、libpfm 等库函数完成，但这些库函数是面向用户层的，不能在内核中直接使用。另一种方法是使用汇编指令直接设置和读取与硬件性能计数器相关的 MSR 寄存器，但由于处理器的硬件性能计数器有限，如 Intel P6 Family 只有2个计数器；AMD K7/K8 系列提供4个计数器，当任务数多于计数器时，就要求计数器被不同的任务分时使用，如此类似，CPU 也是分时被使用，当运行某一个任务时，计数器就为该任务计数，当切换到别的任务时，就为切换之后的任务服务，而将切换下来的任务数据保存，从而实现为每个任务虚拟出一套硬件性能计数器的功

能，称为 Virtual 机制。目前有 2 种常用的硬件性能计数器驱动程序 Perfmon[11]和 Perfctr[12]，测试发现，Perfmon 的兼容性不如 Perfctr，于是选择在 Perfctr 的基础上修改，使内核具备自动监测每个用户线程一个或几个事件的功能。

主要的修改如下： 在内核 copy_thread 函数中添加 vperfctr 结构的初始化，将计数事件设置为 "Retired Instructions"；驱动中使用 kmalloc()函数为 vperfctr 结构开辟的内存空间开销太大，改用 slab 高速缓存；将部分声明为 static 的静态函数改为全局函数；在进程描述符中加入成员变量“IPC”，并在每次任务切换时更新上一个任务的 IPC 值。在每一个时间片结束时，可以得到当前时间片的 IPC，记为 $I_{cur\_timeslice}$，在 vperfctr 数据结构中还有一个累计的 PMC 和 TSC，这 2 个值相除可以得到该任务当前时间片之前的平均 IPC，记为 I_{old_avg}，新的平均 IPC，记为 I_{new_avg}，由下式计算得出。

$$I_{new\_avg} = (1-\alpha)I_{old\_avg} + \alpha I_{cur\_timeslice}$$

计算得到的新 IPC 作为下一个时间片的 IPC 预测值，用于任务迁移决策依据。根据时间和空间局部性原理，相邻几个时间片的行为最为接近，因此 α 取值建议为 0.8 左右，α 过大也不好，会使得 IPC 波动太大，导致迁移过于频繁，反而降低性能。

4.2 重新定义核的负载计算公式

核的负载计算是负载均衡的基础，负载均衡就是周期性地检查调度域内不同调度组之间以及不同调度域之间的负载是否存在较大差异来决定是否实施任务迁移，因此，准确性十分关键。在默认调度器中，每个核的负载是由核内运行队列中所有任务的负载累加得到的，每个调度组的负载是由该调度组中所有核的负载累加得出的。在异构环境下，时钟频率高的核计算能力强，理应按照一定比例承担更多的负载。 对于只有频率不同的异构多核处理器而言，本文暂把频率作为主要参数，将快核的频率因子定义为 1，慢核的频率因子等于快核相对于慢核频率的倍数 F 乘以一个比例系数，系数的作用是调节 cache、访存延迟等其他因素对性能的影响。

在内核中，定义了一个全局数组 amp_factor[NR_CPUS]存放每个 CPU 的频率因子和其是否为最大/最小频率 CPU 的标志位，修改了 2 个函数 inc_raw_weighted_load 和 dec_raw_weighted_load。其作用是计算运行队列的负载，当有任务 P 进入（或离开）某一个运行队列时，该运行队列的负载随之增加（或（减少）β，β 由原来直接等于任务 P 的负载改为任务 P 的负载乘以该运行队列的频率因子。

除了修改负载的计算公式，还要从硬件上把核的频率真正降下来以实现物理上的异构。通常使用内核中已经提供的 cpufreq 驱动，通过 sysfs 对频率进行修改。amp_factor 随 CPU 频率的变化而变化，为保证频率因子的准确性，在每个时钟中断使用 cpufreq 提供的接口更新 amp_factor。好处在于，如果系统处于同构多处理器环境中，则负载均衡策略不会发生任何改变；一旦处于异构环境，调度器会按照新的 amp_factor 重新计算调度组的负载能力。

4.3 程序行为感知的任务迁移规则

根据新的负载计算公式，当出现负载不均衡时，调度器开始寻找最忙的调度组，接着从最忙的调度组中寻找最忙的核，然后开始寻找待迁移任务。默认顺序是按照先过期队列后活动队列，先高优先级后低优先级，先队尾后队头的顺序选择符合迁移条件的任务并进行迁移，直至达到了待迁移任务数或待迁移负载的最大限制或不再有满足条件的待迁移任务为止，基本算法如下。

算法 1　修改后的任务迁移算法 move_tasks (kernel/sched.c)

从过期队列开始遍历，如果过期队列已经遍历完，则遍历活动队列。

从高优先级开始遍历，直到所有优先级全部遍历，或达到任务迁移数后退出。

如果某一个优先级存在任务列表，则从该优先级链表尾遍历任务。选择的任务必须允许迁移（非 cache-hot 或 running），且满足以下条件，否则，继续遍历下一优先级，寻找任务。

如果目标核为快核，则选取该优先级链表 IPC 最大的任务，次大的任务，…，直到达到了待迁移任务数或待迁移负载的最大限制。

如果目标核为慢核，则选取该优先级链表 IPC 最小的任务，次小的任务，…，直到达到了待迁移任务数或待迁移负载的最大限制。

修改后的任务迁移算法与原有的算法总体框架保持一致，稍微不同的是，原有的调度算法直接从链表尾选择任务，修改后的调度算法改为选择 IPC 最大或最小的任务。要选择 IPC 最大或最小的任务，有 2 种方法，一种是遍历，另一种是

事先排好序。如果每一个任务进入某一个新的运行队列都要排序，开销会很大，因为这不仅发生在任务迁移时，也发生在任务创建以及任务切换时。本文选择通过遍历选择 IPC 最大或最小任务，因为任务迁移的频次要远远小于任务的切换次数，其次，这种算法的复杂度是 O(n)，如果任务数量不多，开销并不大，如果很大，可考虑当任务数量低至某一阈值时再进行迁移，因为当某个核上的可运行任务数很多时，再迁移新的任务并不能带来吞吐量的增加。

5 实验环境与数据分析

5.1 异构多核平台的构建

由于目前还没有商用的单指令集异构多核处理器，只能通过间接方法实现。一种是基于模拟器，模拟器的优点是配置灵活，包括核的数量，时钟频率和 Cache 大小，缺点是速度慢，调试效率低，不能完全真实反映硬件系统；另一种是基于现有的同构多核处理器。Intel 实验室使用公司内部的专用工具实现了真正意义上的异构[9]，其他研究组只能通过 DVFS 调节频率实现性能异构，本文也是如此。借助 Linux 内核都的 cpufreq 驱动可以让具有 EIST(enhanced Intel speedstep technology)或 DBS(demand based switching)特征的 Intel 多核处理器和具有 PowerNow 技术的 AMD 多核处理器实现变频。但研究过程中发现了一个奇怪的现象，Intel(R) Core(TM)2 Duo CPU T5550 处理器在 Linux 2.6.34 内核中可以实现将其中某一个核变频，/proc/cpuinfo 也显示出频率的差异，但在不同频率下运行任意一个 benchmark 程序，执行时间却没有差异。进行源代码调试后发现，/proc/cpuinfo 显示的频率并非来自代表 cpu 当前真实频率的寄存器 MSR_IA32_PERF_STATUS，而是读取的一个临时全局变量。后在 Linux 2.6.21 内核上测试，验证了此问题的存在，与 2.6.34 内核不同的是，无论更改哪一个核的频率，另一个核的频率也自动被更改。以上现象说明，虽然 Intel 很多处理器都支持 DVFS，但大部分处理器不支持单个核的频率调节，部分文献提到 Xeon E5440、X5335 等可以实现单个核的频率调节，由于缺少这样的 CPU 而无法进行实际验证。本文是在 Dell PowerEdge 2970 服务器上完成的，包含 2 颗 AMD Opteron 2350 四核处理器，共享的三级片上高速缓存 2M，最大频率 2GHz。

5.2 实验数据分析

1) 程序行为分析开销

程序行为在线分析最大的挑战就是开销不能过大。本文使用硬件性能计数器，计数是在程序执行过程中由硬件同步完成的，开销几乎可以忽略。另一开销来源于修改后的调度算法，根据修改或添加的代码，确定额外开销最可能来源于遍历优先级链表时寻找 IPC 最大或最小的任务。随机选取了 SPEC CPU 2006 中 8 个程序，修改后的调度器相比原有调度器，程序的执行时间并无明显增加，如表 1 所示。

表 1 单一基准测试程序在有无 PMU 的运行时间比较

基准测试程序	异构多核调度器/s	原有调度器/s	时间差异
bwaves	22.24	22.23	0.01
gamess	19.17	19.07	0.10
gcc	30.16	30.06	0.09
hmmer	3.49	3.41	0.08
lbm	6.51	6.49	0.03
libquantcm	16.22	16.16	0.06
mcf	48.10	48.07	0.04
sjeng	59.41	59.36	0.05

2) 系统性能验证

在 AMD 处理器核的频率分别为 2GHz 和 1GHz 时，测得的 12 种基准测试程序的加速比如图 3 所示，可以看出，不同的程序虽然都有所加速，但差异较大，最大的加速比达到 1.72，最小的加速比为 1.19。

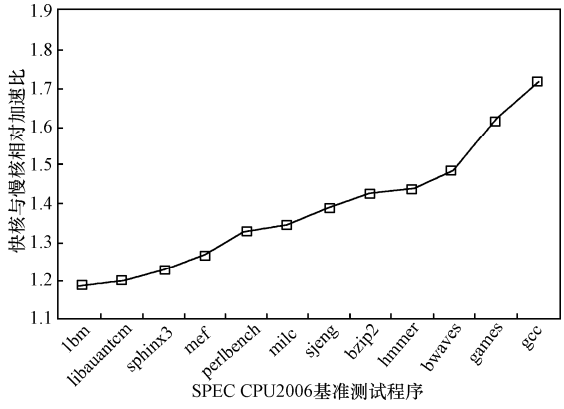

图 3 部分 SPEC CPU2006 程序在 2G 快核与 1G 慢核上的加速比

为了比较异构环境下使用程序行为感知的调度器与原有调度器的性能差异，选择了相对加速比差异较大的 lbm 与 bwaves 作为构造异构工作负载

的基准测试程序，构造了 4 组异构多核环境，分别为 0S1F、0F1S、0F1F2S3S、0F1S2F3S，具体配置及分布如表 2 所示，本文没有测试更多的组合，由于选取的随机性，这 4 种异构多核配置以及工作负载具有一定的代表性。

表 2　4 组异构多核在物理 CPU 上的分布

配置情况	在物理 CPU0 上	在物理 CPU1 上
0S1F	Core0 慢核	Core1 快核
0F1S	Core0 快核	Core1 慢核
0F1F2S3S	Core0 快核 Core2 慢核	Core1 快核 Core3 慢核
0F1S2F3S	Core0 快核 Core2 快核	Core1 慢核 Core3 慢核

当为 0S1F 和 0F1S 时，同时运行 2 个 bwaves 程序和 4 个 lbm 程序，当为 0F1F2S3S、0F1S2F3S 时，同时运行 4 个 bwaves 程序和 8 个 lbm 程序，分别使用原有调度器和程序行为感知调度器的完成时间如图 4 所示，结果显示，在 4 种异构环境中，程序行为感知调度器运行的性能要高于原有内核调度器 5% ~20% 左右。性能的提升幅度不仅取决于调度算法本身，而且与负载的异构程度相关，Bias schdeduling 算法在负载差异很大时，性能平均提升 11% 左右[9]，本文平均值也接近该值，与之不同的是，本文使用的计数器要少。另外一些文献所提到的算法是在仿真环境或用户级模拟器上测试的，本文的算法是在内核中直接实现的，无法进行直接比较。

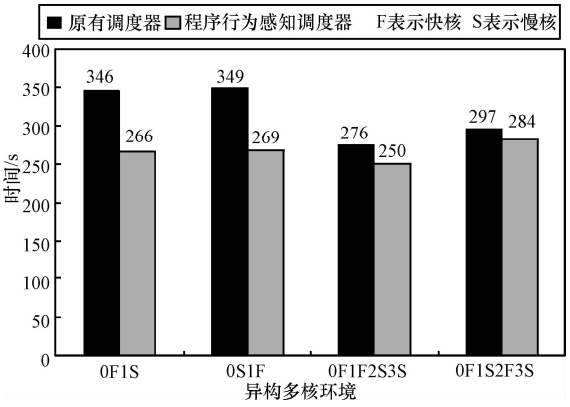

图 4　程序行为感知调度器与原有调度器的性能比较

3）任务分布

为了验证修改后的负载均衡策略的任务迁移是否为所预期的效果，构造了一组人工测试集，包含 2 个程序，分别代表 cpu-intensive 程序和 mem-intensive 程序，IPC 基本恒定为 1.9 和 0.9，各 5 个拷贝。原因在于，SPEC CPU 2006 基准测试程序是通过 shell 脚本触发执行的，初始阶段运行的是 runspec、specperl、sepcinvoke 等命令，不容易观察。人工测试程序虽然行为单一，不具有复杂基准测试程序的代表性，但仅用于观察任务分布还是可以的。为了使效果清晰，只使用了 2 个核，core0(2GHz) 和 core1(1GHz)，分别位于物理 CPU0 和物理 CPU1 上。任务分布变化（如图 5~图 8 所示）显示了负载均衡过程，可以看出，core0 上的任务数绝大多数时候要多于 core1，当 1m 11s 时，一个 mem-intensive 程序从 core0 迁移到 core1，符合向慢核优先迁移访存密集型程序的规则。测试过程中发现，当只有

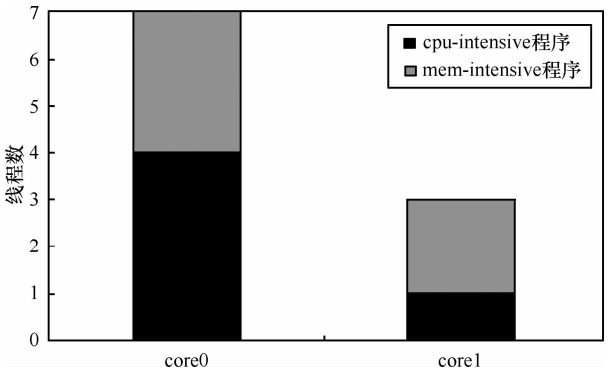

图 5　人工测试集运行 5s 后的分布情况

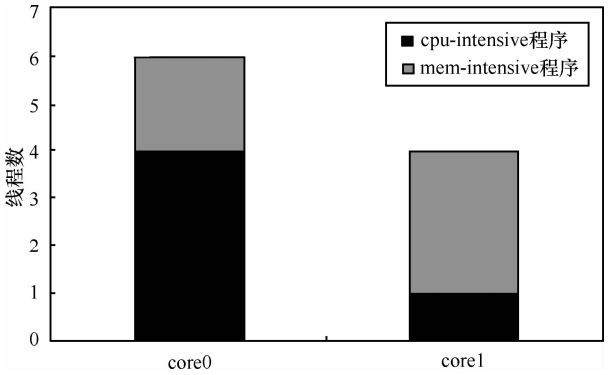

图 6　人工测试集运行 1m 11s 后的分布情况

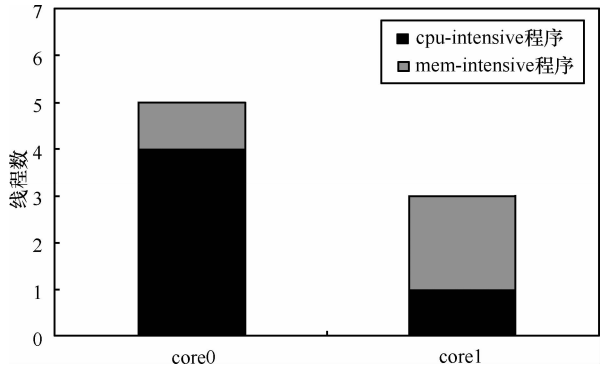

图 7　人工测试集运行 6m 38s 后的分布情况

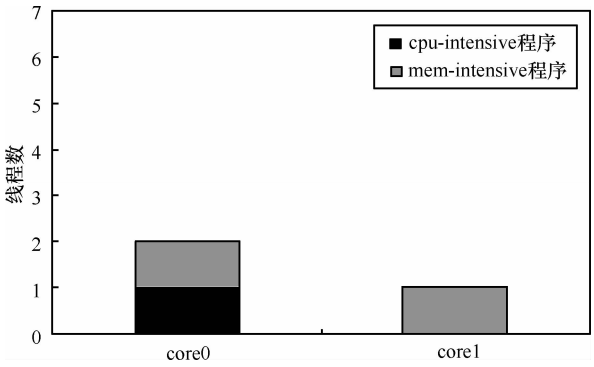

图8 人工测试集运行6m44s后的分布情况

2个程序运行时，并不能保证IPC大的任务一定在快核上运行，原因有二，由于采取的是在线分析，线程刚创建时还没有实际的IPC值，只能按照传统的规则将任务分配到最闲调度组中最闲的核上，很有可能出现IPC大的任务被分配到慢核上，为了保证负载均衡，下一个任务即使IPC小也将分配到快核上；当测试程序数量很少时，迁移效果并不明显，因为很难达到迁移条件。

6 结束语

本文在真实软硬件环境下实现了一个程序行为感知的异构多核调度器，初步测试表明达到了预期的设计目标，开销小，算法简洁。本文提出的算法没有过份强调快核优先，也没有使用阈值来区分程序类别。实际上，cpu-intensive程序或mem-intensive程序都是相对的，阈值很难选取。有的调度算法使用相对值作为metric，这也十分困难，因为要计算相对值，至少要同时得到2个或多个值而不得不采取试探运行的方法，显然在内核中这种开销是无法接受的，本文使用IPC作为metric，在DVFS实现的异构环境下还是可行的，获取容易且不需要试探运行，下一步拟考虑将ALU操作数与访存数的比值作为metric，在程序分类时可能更准确一些，但采用PMU时，由于同时监测的事件类型有限，而不容易实现。

参考文献：

[1] KUMAR R, FARKAS K I, JOUPPIN P, et al. Single-ISA heterogeneous multi-core architectures: the potential for processor power reduction[A]. Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture[C]. San Diego, USA, 2003.81-92.

[2] KUMAR R, TULLSEN D M, RANGANATHAN P, et al. Single-ISA heterogeneous multi-core architectures for multithreaded workload performance[A]. Proceedings of the 31st International Symposium on Computer Architecture[C]. 2004.

[3] FEDOROVA A, SAEZ J C, SHELEPOV D, et al. Maximizing power efficiency with asymmetric multicore systems[J]. Communications of the ACM. 2009, 52(12):48-57.

[4] KUMAR R, TULLSEN D M, JOUPPIN P. Core architecture optimization for heterogeneous chip multiprocessors[A]. Proceedings of the 15th international conference on Parallel architectures and compilation techniques[C]. WA, USA, 2006.23-32.

[5] SHELEPOV D, FEDOROVA A. Scheduling on heterogeneous multicore processors using architectural signatures[A]. Proceedings of the Workshop on the Interaction between Operating Systems and Computer Architecture[C]. Beijing, China, 2008.

[6] SHELEPOV D, SAEZ ALCAIDE J C, JEFFERY S, et al. HASS: a scheduler for heterogeneous multicore systems[J]. ACM SIGOPS Operating Systems Review, 2009, 43(2):66-75.

[7] SONDAG T, RAJAN H. Phase-guided Auto-Tuning for Improved Utilization of Performance-Asymmetric Multicore Processors[R]. 2009.

[8] LIT, BAUMBERGER D, KOUFATY D A, et al. Efficient operating system scheduling for performance-asymmetric multi-core architectures[A]. Proceedings of the 2007 ACM/IEEE Conference on Supercomputing[C]. Nevada, USA, 2007.1-11.

[9] KOUFATY D, REDDY D, HAHN S. Bias scheduling in heterogeneous multi-core architectures[A]. Proceedings of the 5th European conference on Computer systems[C]. Paris, France, 2010.125-138.

[10] SAEZ J C, PRIETO M, FEDOROVA A, et al. A comprehensive scheduler for asymmetric multicore systems[A]. Proceedings of the 5th European conference on Computer systems[C]. Paris, France, 2010.139-152.

[11] ERANIAN S. Perfmon: Linux performance monitoring for IA-64[EB/OL]. http://www.hpl.hp.com/research/linux/perfmon, 2003.

[12] PETTERSSON M. Perfctr: Linux Performance Montioring Counters Driver[R]. Technical Report, Computing Science Department, Uppsala University, 2005.

作者简介：

徐远超（1975-），男，湖北武汉人，中国科学院计算技术研究所博士生，首都师范大学讲师，主要研究方向为多核操作系统、异构计算、云计算、软件形式化验证。

张志敏（1963-），男，浙江仙居人，硕士，中国科学院计算技术研究所研究员，主要研究方向为高性能SoC。

孙卫真（1963-），男，上海人，首都师范大学副研究员，主要研究方向为操作系统、信号处理。